Precise Semantics for Uncertainty Modeling (PSUM)

Version 1.0
Beta - 1

OMG Document Number:	ptc/2023-03-01
Date:	March 2023
Specification URL:	https://www.omg.org/spec/PSUM/1.0/

This OMG document replaces the submission document (ad/22-12-01). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org until September 29, 2023.

Pending issues for this specification may be viewed on the OMG issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in December 2023. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

DISCLAIMER OF WARRANTY

RESTRICTED RIGHTS LEGEND

TRADEMARKS

COMPLIANCE

Contents

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML®(Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Meta-model); and industry-specific standards for dozens of vertical markets. More information on the OMG is available at https://www.omg.org.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at: http://www.omg.org/spec

Specifications are organized by categories; listing selected major categories below:

Fundamental Information Modeling Technologies

- Unified Modeling Langue (UML)
- Systems Engineering Modeling Language (SysML)
- Interface Definition Language (IDL)

Domain-Specific Technologies

- Business and Enterprise Modeling
- Industrial Engineering and Manufacturing
- Space, Control and Transportation Technologies
- Robotics
- Software and Systems Modernization
- Information Security

- Healthcare
- Retail

Middleware and Real-Time Technologies

- Common Object Request Broker (CORBA)
- Data Distribution Service (DDS)
- Middleware Services
- Real-Time and Embedded Systems
- Signal Processing and Communication

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
9C Medway Road, PMB 274
Milford MA 01757
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

Courier - 10 pt. Bold: Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page https://www.omg.org, under Documents, Report a Bug/Issue.

1 Scope

The purpose of this standard is to specify the Precise Semantics for Uncertainty Modeling (PSUM) version 1.0 to build the foundation for developing uncertainty modeling solutions, to guide the implementation of uncertainty modeling tools, to provide the basis for developing training materials and resources in the application of uncertainty modeling, and to serve as the cornerstone of proposing future revisions and extensions of uncertainty modeling solutions.

PSUM v1.0 is an uncertainty modeling metamodel, which 1) captures uncertainty and its related concepts, and 2) enables measurements of uncertainty and uncertainty-related concepts. PSUM v1.0 is intentionally designed to be generic such that it acts as the foundation for developing other modeling languages, e.g., integrating with UML or SysML for achieving specific modeling purposes and user applications, and being specialized for developing domain-specific uncertainty modeling solutions.

We will define the PSUM metamodel using the MOF meta-modeling language, to be consistent with other models defined by OMG. Therefore, it will have a standard textual representation in XMI, which can be used by software to check its conformance to the PSUM specification.

This page intentionally left blank

2 Conformance

PSUM defines a metamodel for defining and representing uncertainty and uncertainty related concepts, associating to SMM for quantifying uncertainty and other measurable concepts such as belief. To be PSUM compliant, an implementation must provide: 1) the capability to generate XMI documents based on the PSUM XMI schema from an existing model of the tool; and 2) the capability to import a model that is based on the PSUM XMI schema into the tool.

This specification defines two levels of conformance:

- *Level 1* requires full implementation of PSUM except for the *Belief* package. In other words, one can start directly from the *Uncertainty* package to specify, characterize and measure uncertainties without specifying *Belief* and *BeliefStatement*.
- *Level 2* requires full implementation of PSUM including the *Belief* package, implying that an implementation needs to cover the complete PSUM metamodel.

Implementations of both levels need to include SMM, because PSUM refers to SMM for quantifying *Uncertainty* and other measurable elements. Quantification is an important aspect of uncertainty modeling.

Implementations of both levels can optionally include SACM. This is because PSUM defines *Evidence* and classifies it into four types (*EvidenceType*), which might be sufficient in some uncertainty modeling contexts.

This page intentionally left blank

3 References

3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

[SACM] Structured Assurance Case Metamodel (SACM), version 2.2, OMG Specification
https://www.omg.org/spec/SACM/2.2

[SMM] Structured Metrics Metamodel (SMM), version 1.2, OMG Specification
https://www.omg.org/spec/SMM/1.2

[SysML] OMG Systems Engineering Modeling Language (SysML), version 1.6, OMG Specification
https://www.omg.org/spec/SysML/1.6

[UML] Unified Modeling Language (UML), version 2.5.1, OMG Specification
https://www.omg.org/spec/UML/2.5.1

[XMI] XML Metadata Interchange (XMI), version 2.5.1, OMG Specification
https://www.omg.org/spec/XMI/2.5.1

[ISO 31000] Risk management, International Standard
https://www.iso.org/iso-31000-risk-management.html

3.2 Non-normative References

None.

This page intentionally left blank

4 Terms and Definitions

None.

This page intentionally left blank

5 Symbols

None.

This page intentionally left blank

6 Additional Information

(informative)

6.1 Guide to this Specification

6.1.1 Document Conventions

All clauses of this document are *normative* unless explicitly marked “(informative)”. The marking “(informative)” of a particular clause also applies to all contained sub-clauses of that clause.

Specification text is written in normal Roman font.
Model elements are referenced within the text using *Italic font*.
OCL constraints are written in `Code font`.

6.1.2 Clause Structure

Clause 7 is an informative overview of the principles of uncertainty. Clause 8 defines the abstract architecture of the PSUM metamodel. Clause 9 introduces the foundational elements of the PSUM metamodel. Clause 10 defines the Belief package. Clause 11 defines the Uncertainty concepts of PSUM. Clause 12 define the Measurement model used by PSUM. Clause 13 defines the PSUM Evidence model as derived from SACM. Annex A provides examples how to use PSUM for uncertainty modeling.

6.2 Intellectual Property Rights

The Precise Semantics for Uncertainty Modeling specification is available to everyone under the OMG Non-Assertion Covenant Intellectual Property Rules, and under the copyrights listed on the first page of this document.

6.3 Changes to Existing OMG Specifications

This specification does not require or impose changes to excising OMG specifications.

6.4 Acknowledgments

The following organizations submitted this specification:

- 88solutions Corporation
- Simula Research Laboratory
- SOFTEAM (Docaposte Group)

The following organizations contributed to this specification:

- Thematix Partners LLC
- University of Duisburg-Essen
- University of Málaga
- Helmut Schmidt University

This page intentionally left blank

7 Background and Rationale

(informative)

Uncertainty has been studied in various fields, such as philosophy, physics, statistics, and finance, to describe a situation of lacking knowledge about the state of a system and/or potential future outcome(s) [1]. In the literature, various efforts (e.g., [2, 3, 4, 5]) have been made to understand and classify uncertainty for various purposes (e.g., supporting decision-making), from different perspectives (e.g., ethics), and in diverse domains (e.g., healthcare).

As defined in [6], *uncertainty* means "the lack of confidence (i.e., knowledge) about the timing and nature of inputs, the state of a system, a future outcome, as well as other relevant factors". In the ISO 31000 standard [1], *uncertainty* is defined in the context of Risk Management as "the state, even partial, of deficiency of information related to, understanding or knowledge of an event, its consequence, or likelihood."

It is therefore important to *face*, *understand*, and explicitly *specify/model* uncertainty in system/software engineering. By "facing" uncertainty, we mean that it is important to identify uncertainties explicitly, analyze them, and make an effort to deal with them in all phases of development, including, for instance, evaluating effects of uncertainties, reasoning about how uncertainty propagates (with the goal to reduce it when possible), or at least constraining and alleviating its consequences. To "understand" uncertainty, it is necessary to clearly and precisely define it and all its relevant concepts. By "specifying" uncertainty, we mean that, wherever applicable, a precise formulation of uncertainty should be explicitly associated with various development artifacts, such as requirements, analysis, and design models, or test-ready models (to enable the generation of test cases for testing systems/software under uncertainties).

Uncertainty is gradually gaining more and more attention in system/software engineering these days. In particular, complex software-intensive systems, such as Cyber-Physical Systems (CPSs) and self-adaptive systems, typically operate in dynamic and unpredictable environments. A survey of how uncertainty has been specified so far in software models is presented in [7]. In [8], the authors compiled different categorizations of uncertainties in the context of self-adaptive systems. However, a unified and generally applicable conceptual fundamental and common terminology for specifying uncertainty is missing, which also includes a precise elaboration of relationships between uncertainty and other related concepts such as measurements. Specifically, potential applications of uncertainty modeling include: (1) specifying uncertainty requirements in use case models, (2) modeling uncertainties as part of SysML or UML analysis and design models, (3) modeling uncertainties as part of test-ready models for enabling model-based testing, which might be specified with extensions to SysML, UML and/or MARTE and UTP V2.0, (4) modeling uncertainties as part of BPMN models, (5) modeling uncertainties with an independent modeling notation dedicated to uncertainty modeling. Models with explicitly specified uncertainty could then be used as the basis for performing different analyses, such as discovering unanticipated uncertainties or generating other artifacts such as test cases.

This Precise Semantics for Uncertainty Modeling (PSUM) specification covers in particular the following two aspects: (1) Capturing uncertainty and its related concepts; (2) Enabling measurements of uncertainty and uncertainty-related concepts.

This page intentionally left blank

8 PSUM Overview

The PSUM metamodel is composed of three top packages: *Overall*, *Core*, and *Measurement*. The *Core* package is composed of the sub-packages of *Evidence* and *Risk*, *Uncertainty*, and *Belief*. In addition, the *Evidence* package of PSUM is dependent on the SACM specification [9], while the *Measurement* package of PSUM is dependent on the SMM specification [10], as shown in Figure 8.1 given below.

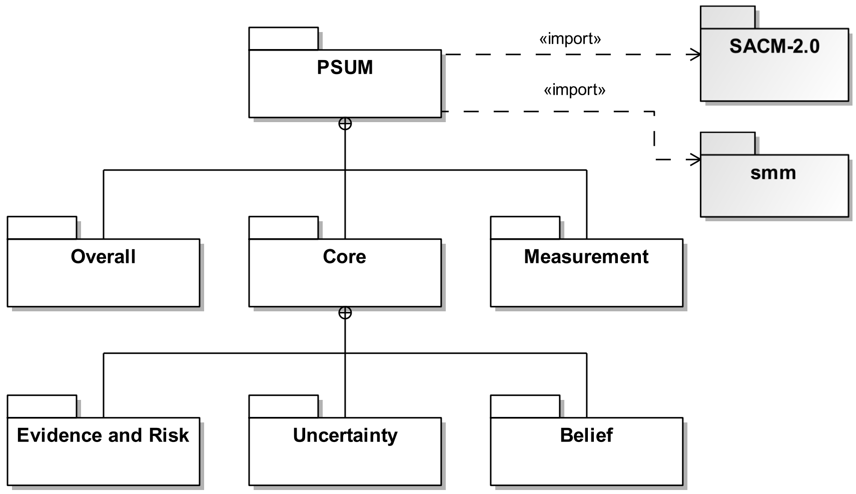

Figure 8.1: PSUM Package Structure

This page intentionally left blank

9 Overall

The *Overall* package defines the PSUM model and its elements (i.e., *PSUMElement*), as shown in the figure below. PSUM elements may have dependencies among them, which can be typed as *Derive*, *Trace*, *Use* and *Refine*. Each PSUM element may optionally be associated with annotations and may also have optional attributes (characterized by tag and value pairs). In addition to ID and description, a PSUM element may also be characterized by a point in time, indicating when the model element was specified.

Also, notice that an association is added in the PSUM metamodel to indicate that measurements of measurable PSUM elements (e.g., uncertainty) are specified with the SMM standard [10].

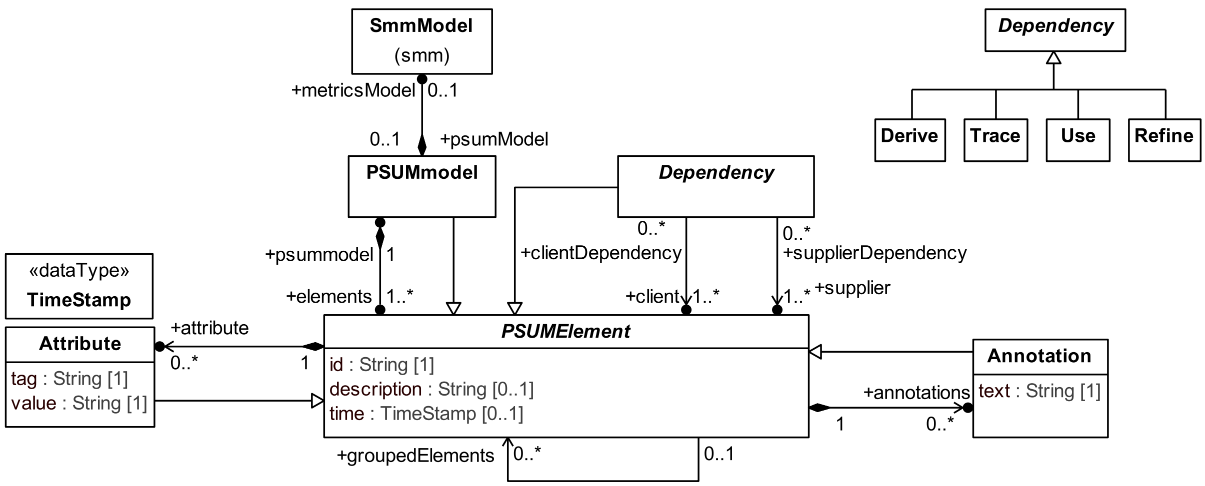

Figure 9.1: PSUM Overall Package

9.1 PSUMelement [Class]

PSUMelement is a constituent of a PSUM model. All the other model elements of the PSUM metamodel specialize *PSUMElement*, though we do not show these specializations in the figures. A *PSUMelement can optionally have a* TimeStamp representing a point in time.

9.2 Dependency [Class]

Dependency represents that a single model element, or a set of model elements, requires other model elements for their specification or implementation. This means that the complete semantics of the client element(s) are either semantically or structurally dependent on the definition of the supplier element(s). [11]

9.3 PSUMmodel [Class]

PSUMmodel consists of a set of PSUM elements and SMM model elements. The latter are subsumed in a *SmmModel* connected to a *PSUMmodel*.

9.4 Attribute [Class]

Attribute allows information to be attached to any model element in the form of a tagged value pair. An associated constraint does not allow attributes to have attributes or annotations attached to them, as formally specified below:

```
context Attribute inv NoAttributeOrAnnotationOnAttribute:
    self.attribute->size()=0 and self.annotations->size()=0
```

9.5 Annotation [Class]

Annotation allows textual descriptions to be attached to any instance of a model element [10]. An associated constraint does not allow annotations to have attributes or annotations attached to them, as formally defined below:

```
context Annotation inv NoAttributeOrAnnotationOnAnnotation:
     self.attribute->size()=0 and self.annotations->size()=0
```

9.6 Derive [Class]

Derive specifies a derivation relationship among model elements that are usually, but not necessarily, of the same type. Such a dependency specifies that the client may be computed from the supplier [11].

9.7 Trace [Class]

Trace specifies a trace relationship between model elements or sets of model elements that represent the same concept in different models [11].

9.8 Use [Class]

Use is a dependency that one *PSUMElement* requires another *PSUMElement* (or set of *PSUMElements*) for its full implementation or operation. The use dependency does not specify how the client uses the supplier other than the fact that the supplier is used by the definition or implementation of the client [11].

9.9 Refine [Class]

Refine specifies a refinement relationship between model elements at different semantic levels, such as analysis and design [11].

10 Belief

The PSUM belief metamodel package captures *Belief*s made by *BeliefAgents*. A belief is composed of at least one belief statement. A belief statement can be associated with a set of uncertainties. Both a belief and an uncertainty can be associated with indeterminacy sources. A belief is made based on *PersonalExperience* and/or *Evidence*. A belief itself can be a basis to support the proposal of another belief.

Between *BeliefStatement* and *Uncertainty*, we capture *UncertaintyTopic*, which is defined to relate the uncertainty to its subject (i.e., the associated *BeliefStatement* about which exists uncertainty). Furthermore, a *BeliefStatement* can include more than one associated *UncertaintyTopic*s. An *UncertaintyTopic* can be associated with one or more *BeliefStatement*s. For example, the "wind speed" *UncertaintyTopic* can be discussed in multiple *BeliefStatement* elements. Also, the same *UncertaintyTopic* can be discussed in more than one *BeliefStatement* described using different languages (e.g., graphical modeling languages such as UML, or natural-language statements in English).

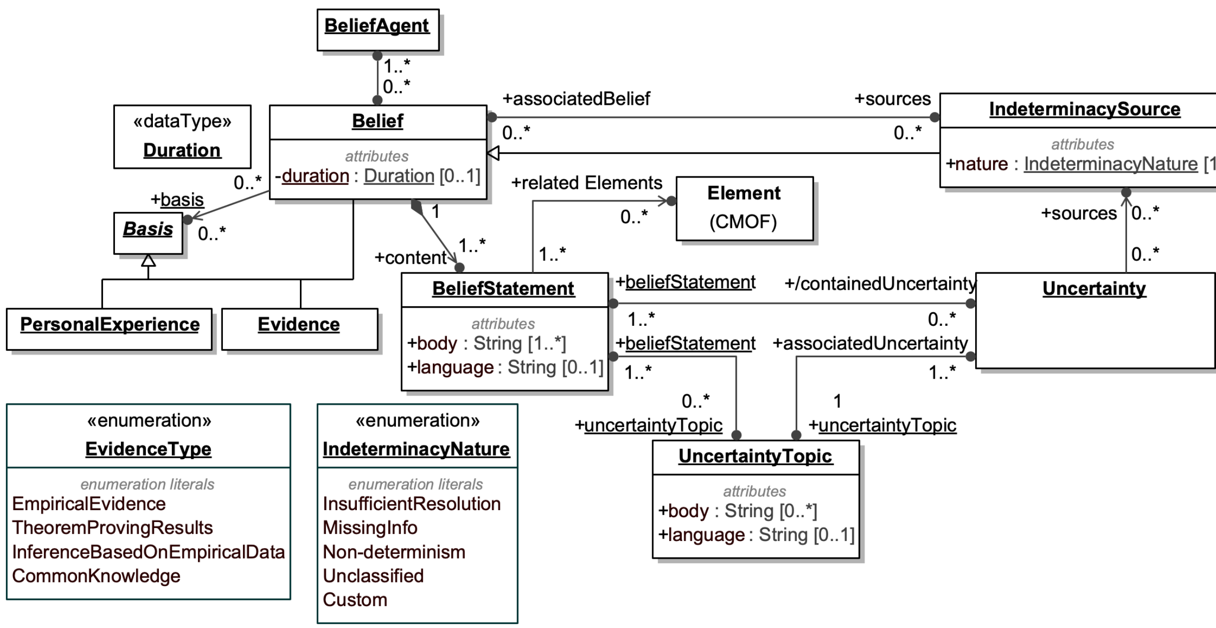

Figure 10.1: PSUM Belief Model

10.1 Belief [Class]

Belief is an implicit subjective explanation or conceptualization of some phenomena or notions, which is held by a *BeliefAgent*. Note that the term “phenomena” is intended to cover aspects of objective reality, whereas “notion” covers abstract concepts, such as those encountered in mathematics or philosophy [6].

In PSUM, implicit *Belief* is distinguished from *BeliefStatement*, which is an explicit representation of (parts of) a *Belief*. Beliefs are handled as explicitly specified content; therefore, a belief must have at least one *BeliefStatement*. The following two constraints are applied to *Belief*:

The indeterminacy sources associated to an uncertainty should be included in the set of indeterminacy sources of the corresponding belief:

```
context Belief inv ConsistentSources:
     self.sources->includesAll(self.content.containedUncertainty.sources)
```

A belief cannot form part of the basis for itself:

```
context Belief inv NotSelfBasis:
     self.basis->closure(b|b.oclAsType(Belief).basis)->excludes(self)
```

A belief can optionally have its *duration* specified, which indicates the time during which the belief is considered valid.

10.2 BeliefAgent [Class]

BeliefAgent is a physical entity that holds (i.e., owns) one or more beliefs about phenomena or notions associated with one or more subject areas. This could be a human individual or group, an institution, a living organism, or even a machine such as a computer [6]. Crucially, a belief agent is capable of actions based on its beliefs [6].

10.3 BeliefStatement [Class]

BeliefStatement is a concrete, tangible representation of (parts of) a belief, specified according to the syntactic rules of a specific language. A *BeliefStatement* can have one or many bodies, which share the same set of Uncertainty. For example, the same content can be specified in different natural languages; the same content ("the door is open") can be specified as different belief statements in the same language ("the door is open" and "the door is 3 centimeters to the door frame.").

10.4 Uncertainty [Class]

Uncertainty is the state of deficiency (e.g., lack or partiality) of information or knowledge required to understand the content, consequence or likelihood of an *UncertaintyTopic* existent in a *BeliefStatement*.

10.5 UncertaintyTopic [Class]

UncertaintyTopic relates an *Uncertainty* to its subject (i.e., the associated *BeliefStatement*).

Role *containedUncertainty* of the association between *BeliefStatement* and *Uncertainty* is a derived role. Uncertainties contained in a *BeliefStatement* can be derived through *UncertaintyTopic*:

```
context BeliefStatement::containedUncertainty : Set(Uncertainty) derivedSet:
     self.uncertaintyTopic.associatedUncertainty->asSet()
```

10.6 IndeterminacySource [Class]

IndeterminacySource is a situation where the information required to ascertain the validity of a belief statement is indeterminate in some way, resulting in uncertainty being associated with that statement [6].

10.7 IndeterminacyNature [Enumeration]

IndeterminacyNature specifies different kinds of indeterminacy that characterize an *IndeterminacySource*. [6]

- *InsufficientResolution* denotes that the information available about the phenomenon in question is not sufficiently precise;
- *MissingInfo* denotes that the full set of information about the phenomenon in question is unavailable at the time when the statement is made;
- *Non-determinism* denotes that the phenomenon in question is either practically or inherently non-deterministic;
- *Unclassified* denotes indeterminate indeterminacy;
- *Custom* is a user-defined indeterminacy.

10.8 Basis [Class]

Basis is the basis on which a *BeliefAgent* makes a *Belief*.

10.9 PersonalExperience [Class]

PersonalExperience is also named as anecdotal evidence, which is defined as testimony that something is true, false, related, or unrelated based on isolated examples of someone's personal experience.

10.10 Evidence [Class]

Evidence is either an observation or a record of a real-world event occurrence or, alternatively, the conclusion of some formalized chain of logical inference that provides information that can contribute to determining the validity (i.e., truthfulness) of a *Belief*. It is inherently an objective phenomenon representing something that actually happened, which implies that we exclude the possibility of counterfeit or invented evidence. Nevertheless, although *Evidence* represents objective reality, it need not be conclusive in the sense that it removes all doubts (*Uncertainty*) about a *BeliefStatement*.

10.11 EvidenceType [Enumeration]

EvidenceType consists of four literals:

- *EmpiricalEvidence* representing information acquired by observation or experimentation;
- *TheoremProvingResults* denoting evidence collected from results of theorem proving;
- *InferenceBasedOnEmpiricalData* representing Information collected from inference based on empirical data;
- *CommonKnowledge* denoting knowledge known to everyone or nearly everyone.

This page intentionally left blank

11 Uncertainty

This part of the PSUM metamodel describes concepts that characterize uncertainty, via the abstract metaclass *UncertaintyCharacteristic* and its subtypes: *UncertaintyPerspective*, *Pattern*, and *Effect*. Uncertainty is further characterized by its attributes: kind, nature, and reducibility. Classifications of uncertainty based on these three attributes can be described using the three enumerations: *UncertaintyKind*, *UncertaintyNature*, and *ReducibilityLevel*.

In addition, this part of the metamodel captures the evolution of uncertainty. For this, the metamodel defines *UncertaintyHistory*. The history of an uncertainty captures a time series of changes made to its characteristics. Often, uncertainty measurements are changed over time. This is captured via the association from *UncertaintyHistory* to Measurement of SMM.

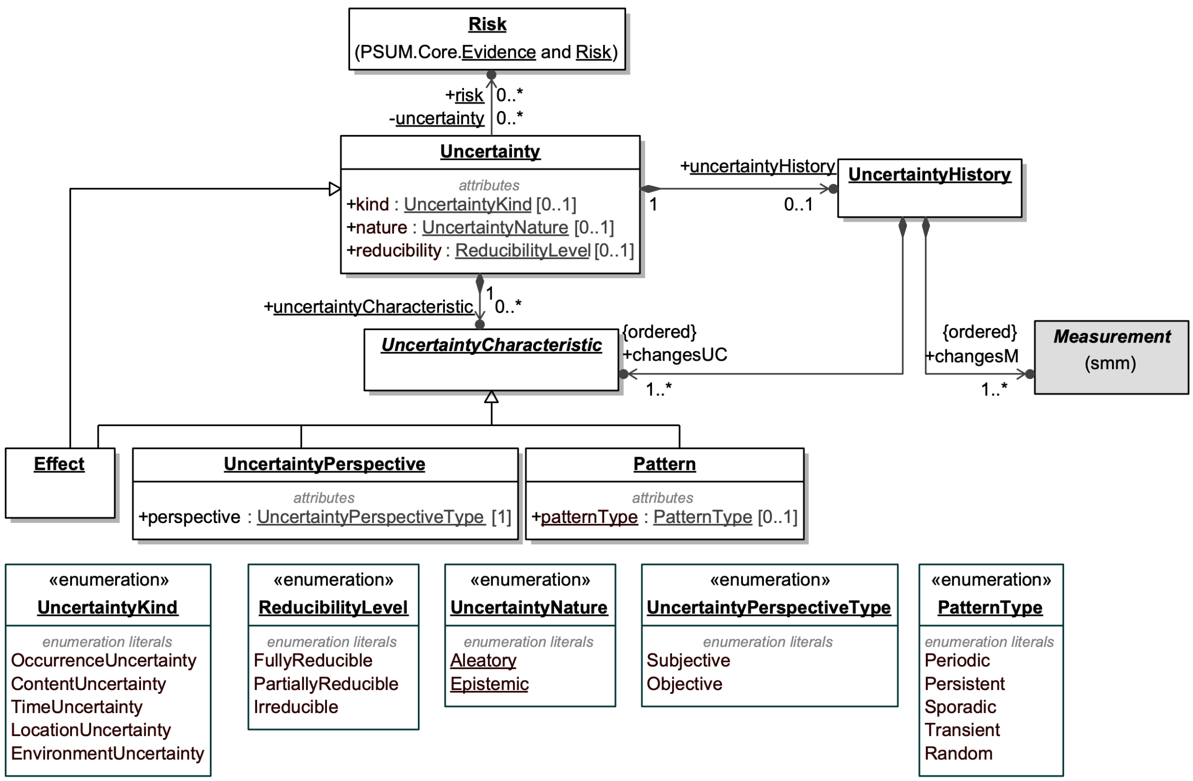

Figure 11.1: PSUM Uncertainty Model

11.1 Uncertainty [Class]

Uncertainty is the state of deficiency (e.g., lack or partiality) of information or knowledge required to understand the content, consequence or likelihood of an *UncertaintyTopic* existent in a *BeliefStatement*.

The following integrity constraints apply to class *Uncertainty*.

Aleatory uncertainty is irreducible, in that there will always be variability in the underlying variables.

```
context Uncertainty inv AleatoryIsIrreducible:
    self.nature = UncertaintyNature::Aleatory
        implies self.reducibility=ReducibilityLevel::Irreducible
```

Epistemic uncertainty is reducible, in that additional information or knowledge may reduce it.

```
context Uncertainty inv EpistemicIsReducible:
    self.nature = UncertaintyNature::Epistemic implies
```

```
    (self.reducibility = ReducibilityLevel::PartiallyReducible or
    self.reducibility = ReducibilityLevel::FullyReducible)
```

Only occurrence uncertainties can have one pattern.

```
context Uncertainty inv NoPatternForNonOccurrenceUncertainty:
    self.kind <> UncertaintyKind::OccurrenceUncertainty implies
        self.uncertaintyCharacteristic->select(oclIsKindOf(Pattern))->size()=0
```

One may or may not define a pattern for an occurrence type of uncertainty.

```
context Uncertainty inv PatternForOccurrenceUncertainty:
    self.kind = UncertaintyKind::OccurrenceUncertainty implies
        self.uncertaintyCharacteristic->
            select(oclIsKindOf(Pattern))->size()<=1
```

11.2 UncertaintyCharacteristics [Class]

UncertaintyCharacteristics is about information that in some way characterizes the associated uncertainty. It is specialized into the more detailed characteristics *Effect*, *UncertaintyPerspective*, and *Pattern*.

11.3 UncertaintyHistory [Class]

UncertaintyHistory keeps the record of a sequence of changes applied to any *UncertaintyCharacteristic* or *Measurement* of an *Uncertainty*.

11.4 UncertaintyPerspective [Class]

UncertaintyPerspective is an uncertainty characteristic that specifies whether an uncertainty is subjective or objective in terms of the perspective of observing agents.

11.5 UncertaintyPerspectiveType [Enumeration]

UncertaintyPerspectiveType defines the following types:

- *Subjective* uncertainty perspective refers to information existing within some agency derived from observation and/or reasoning by that agency;
- *Objective* uncertainty perspective refers to phenomena or concepts whose existence and nature are independent of any observing agency.

If an uncertainty is fully reducible, then it is possible to transit from subjective to objective when having enough evidence.

11.6 Pattern [Class]

Pattern is an intelligible way in which an uncertainty appears.

11.7 PatternType [Enumeration]

PatternType defines the following types - from [6]:

- *Periodic* pattern occurs at regular intervals of time;
- *Persistent* pattern lasts forever;
- *Sporadic* pattern occurs occasionally;
- *Transient* pattern occurs temporarily;
- *Random* pattern occurs without a definite method, purpose or conscious decision.

11.8 Effect [Class]

Effect captures the result of an uncertainty in a belief statement, i.e., the consequence of misinterpreting the *BeliefStatement* due to uncertainty associated with it. An uncertainty may result in another known uncertainty; therefore, an *Effect* itself can be another *Uncertainty*.

11.9 UncertaintyKind [Enumeration]

UncertaintyKind includes the following types - from [6]:

- *ContentUncertainty* represents a situation whereby a *BeliefAgent* is uncertain about the content expressed in a *BeliefStatement*;
- *EnvironmentUncertainty* represents a situation whereby a *BeliefAgent* is uncertain about the surroundings of a physical system existing in a *BeliefStatement*;
- *GeographicalLocationUncertainty* represents a situation whereby a *BeliefAgent* is uncertain about the geographical location existing in a *BeliefStatement*;
- *OccurrenceUncertainty* represents a situation whereby a *BeliefAgent* is uncertain about the occurrence of events existing in a *BeliefStatement*;
- *TimeUncertainty* represents a situation whereby a *BeliefAgent* is uncertain about time existing in a *BeliefStatement*.

11.10 ReducibilityLevel [Enumeration]

ReducibilityLevel includes the following types:

- *FullyReducible* uncertainty denotes that there is no full certainty, but uncertainty is knowable and can be reduced by collecting additional information until achieving full certainty (no irreducible uncertainty present);
- *PartiallyReducible* uncertainty means that there is no full certainty, but uncertainty can be reduced by collecting additional information;
- *Irreducible* uncertainty means that there is no full certainty, and it cannot be reduced.

11.11 UncertaintyNature [Enumeration]

UncertaintyNature of an uncertainty indicates whether it is of the aleatoric nature or epistemic nature.

- *Aleatory* uncertainty subsumes uncertainty caused by the inherent stochasticity of real-world phenomena;
- *Epistemic* uncertainty is caused by lack of information [12].

This page intentionally left blank

12 Measurement

The measurement part of the PSUM metamodel aims to provide a generic approach to quantify *Belief*, *Uncertainty*, *Risk*, and *IndeterminacySource*, which are all considered as *MeasurableElements* of the PSUM metamodel. A measurable element may be composed of a list of measurable features, which can be accuracy, sensitivity, measurement error, precision, and degree. In addition, a measurable feature is associated with *Measurement* of the SMM specification, through which a measurable element (e.g., the degree of a belief) can be measured, with a specific Measure (defined in the SMM standard [10]).

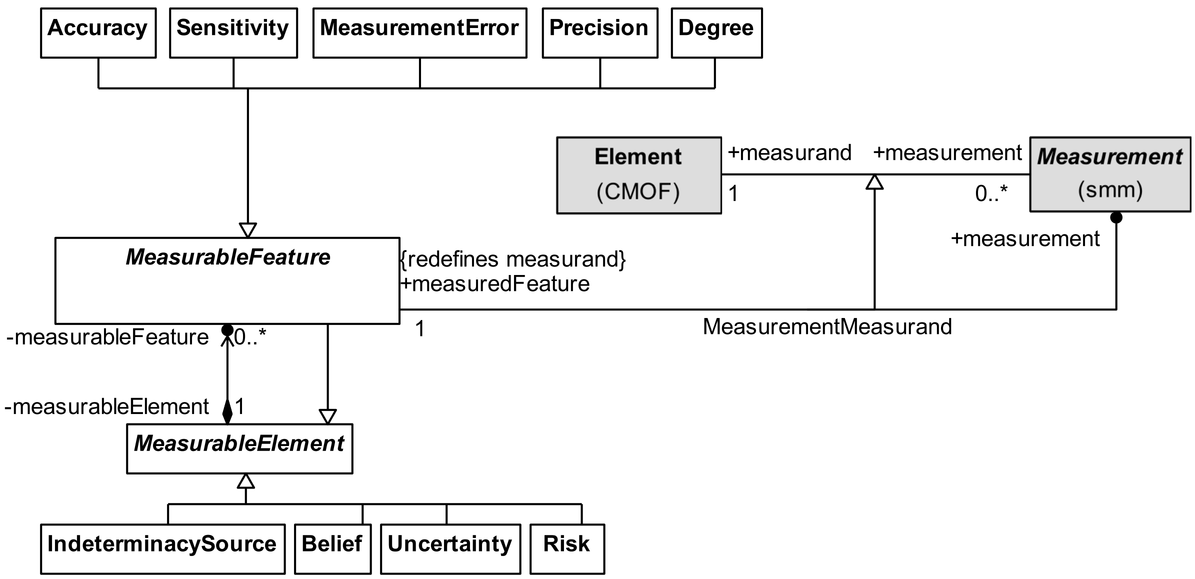

Figure 12.1: PSUM Measurement Package

12.1 MeasurableFeature [Class]

MeasurableFeature of a *MeasurableElement* quantifies it by a *Measurement*, a numerical or symbolic value assigned to the *MeasurableElement* by a *Measure* [10].

12.2 MeasurableElement [Class]

MeasurableElement is a *PSUMElement* that can be measured. A *MeasurableElement* is decomposed by a set of associated *MeasurableFeature* instances, i.e., there might be multiple different aspects that can be measured in order to quantify a *MeasurableElement*.

12.3 Risk [Class]

Risk is the effect of uncertainty on objectives [1].

12.4 Accuracy [Class]

Accuracy measures the closeness of agreement between a quantity value obtained by measurement and the true value of the measurand [13].

12.5 Sensitivit [Class]

Sensitivity is defined as the quotient of the change in an indication of a measuring system and the corresponding change in a value of a quantity being measured [13]. Sensitivity is an absolute quantity, the smallest absolute amount of change that can be detected by a measurement.

12.6 MeasurementError [Class]

MeasurementError is defined as the result of a measurement minus a true value of the *measurand* [13]. Also called "absolute error of measurement", in order to distinguish it from "relative error" which is the measurement error divided by a true value of the measurand [13].

12.7 Precision [Class]

Precision measures the closeness of agreement between quantity values obtained by replicate measurements of a quantity, under specified conditions. Measurement precision is usually expressed numerically by measures of imprecision, such as standard deviation, variance, or the coefficient of variation under the specified conditions of measurement [13].

12.8 Degree [Class]

Degree is one kind of *MeasurableFeature* that can be used to measure *IndeterminacySource*, *Belief*, *Risk* and *Uncertainty*, by indicating their extent, according to either a scale, a numerical value such as a probability or any other measure of subjective uncertainty (e.g., a fuzzy value, a subjective logic opinion).

- Degree of *IndeterminacySource* captures the extent to which there is uncertainty about the *IndeterminacySource*.
- Degree of *Belief* captures the extent to which a belief agent is confident about the validity of the *Belief*.
- Degree of *Risk* captures the severity of a *Risk* associated with uncertainty.
- Degree of *Uncertainty* captures the extent to which the *Uncertainty* persists.

13 Evidence

The PSUM metamodel only defines *Evidence*, as the SACM specification [9] does not define this concept explicitly. Through Evidence, the PSUM metamodel is connected to SACM, specifically its concepts of ArtefactElement and ArtefactElementCitation. In SACM, ArtefactElement is defined as “objective artefacts being offered in support of one or more claims” [9]. In the PSUM metamodel, we define the enumeration EvidenceType, which provides an opportunity to characterize evidence at a very high level. Users are encouraged to associate PSUM to SACM for comprehensive modeling of evidence when needed.

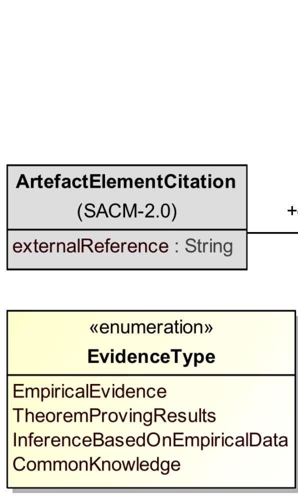

Figure 13.1: PSUM Evidence Model

13.1 Evidence [Class]

Evidence (see its definition in Section 10) is associated with a set of *ArtefactElement* of SACM, which is defined as “objective artefacts being offered in support of one or mode claims”. [9]. See also the definition of *EvidenceType* in Section 10.

This page intentionally left blank

Appendix A Examples of Applying PSUM

(informative)

A.1 Specifying Uncertainty Requirements

This case study is a partial representation of beliefs and uncertainties specified in the use case specification (UCS) presented in [14]. In [14], a methodology, named U-RUCM, was presented to specify uncertainty requirements as part of use case models, which is an integration of the Restricted Use Case Modeling (RUCM) [15] and U-Model [6]. A smart home system implements various safety and security features such as detecting intrusions and fires. One key use case is about a homeowner enabling windows and doors monitoring of her/his home. The UCS of this use case has the basic flow describing the happy path that the monitoring is enabled properly, and the system is running as expected, and there is no intrusion detected. The UCS also contains several alternative flows such that any occurrence of an intrusion should be detected, immediately followed by sending an intrusion notification to the homeowner and the activation of an alarm. With PSUM, we aim to specify uncertainty requirements on this UCS, as described below.

As shown in the object diagram below, we define the *BeliefAgent SRL* (a requirements engineer, a domain expert, or even a smart home development team), who makes all the beliefs presented in the diagram. *SRL* makes a belief on the overall UCS, i.e., *overall UCS belief : Belief*, which has a timestamp ("2021-03-11"), an associated *BeliefStatement* (i.e., a *brief description of the UCS*: "The system monitors the status of windows and doors when Home Owner enables the monitoring function", in which Home Owner is the primary actor the use case), an associated Evidence instance, an associated *IndeterminacySource* instance (i.e., *broken control panel*). In addition, the belief degree is measured via a *MeasurableFeature* instance and an instance of *DirectMeasurement* from SMM, i.e., 80% confidence (in probability) on the belief. The belief statement corresponds to the brief description of the UCS, which summarizes the goal of the UCS, which has its body specified as an English statement: "The system monitors the status of windows and doors when Home Owner enables the monitoring function."

In addition, in the object diagram shown below, we also specify beliefs and belief statements corresponding to the precondition of the UCS (as shown in *precondition of UCS : BeliefStatement*) and one of the steps of the UCS's basic flow (i.e., step 5). The *step 5* belief statement *(belief on step 5 of the UCS's basic flow : Belief*) has its body as "The system enables the monitoring function.", which is also an English statement. The belief statement is associated with an *Uncertainty* (i.e., does *not enable*) via the *enable : UncertaintyTopic*. The uncertainty is characterized by its *Effect*, *UncertaintyPerspective*, *Pattern*, and uncertainty measurement. In addition, in the model, we kept the history of the change made to the uncertainty measurement: from 10% probability to 2% probability.

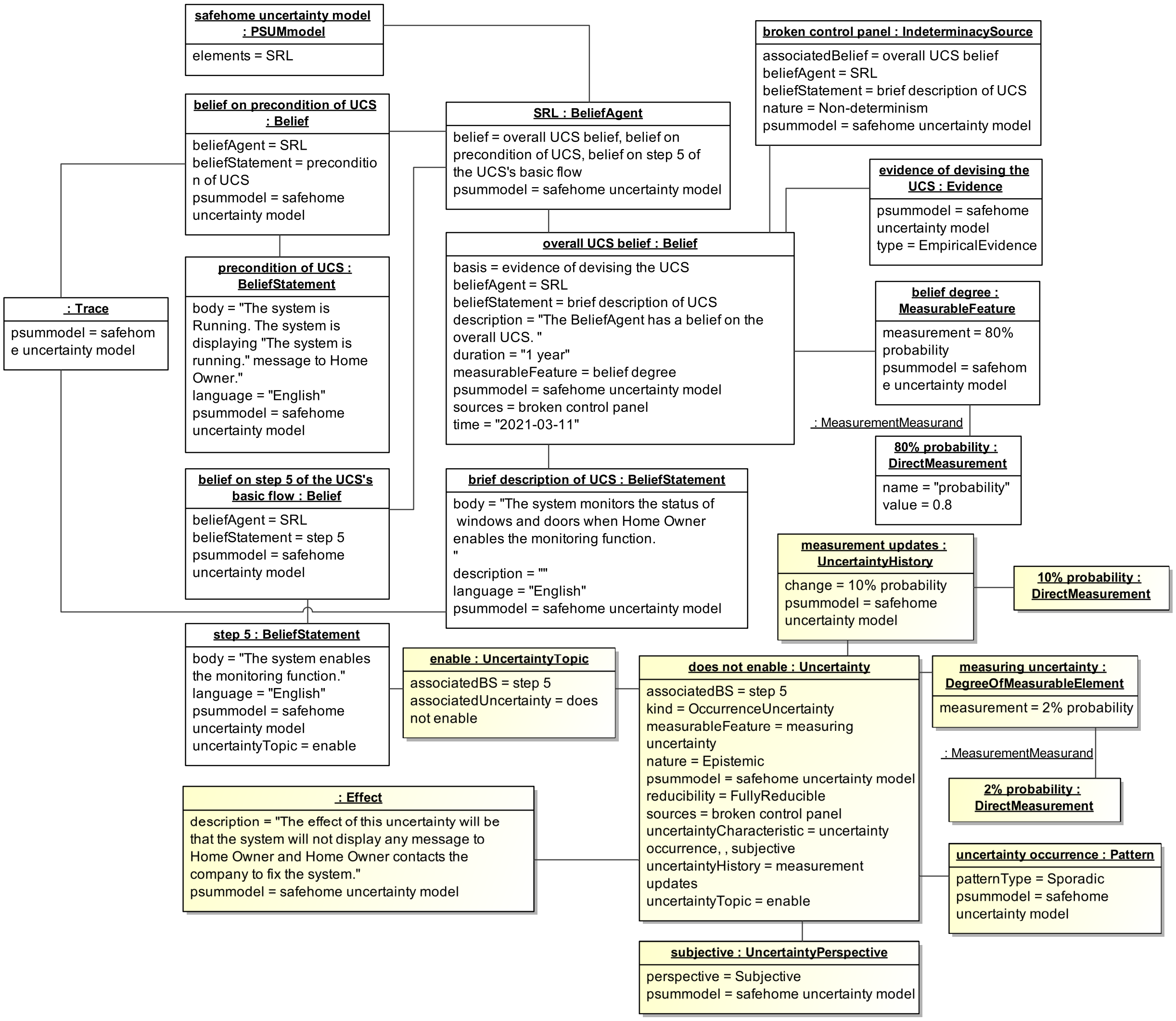

Figure A.1: PSUM Safe Home Example

A.2 Measurement Uncertainties

The following models show how the PSUM metamodel can be used to specify different kinds of measurements and belief uncertainties.

A.2.1 Indicating the Precision of a Measurement

The first example shows the specification of the statement:

"Betty thinks that the length of the room is 3m with a precision of 0.001m."

In this case, the *Belief* is represented by one *BeliefStatement*, which is associated to one *UncertaintyTopic* ("the length of the room"), and to a given Uncertainty, of kind *MeasurementUncertainty* and hence of the *Aleatory* nature. In this case, the measurable feature of interest of the *Uncertainty* is *Precision*, and the model element related to the *BeliefStatement* is the length of the room, another measurable feature. To measure these two measurable features, the

SMM is used. The corresponding SMM elements are shown with unshaded boxes. They assign the concrete values to the measurable features by specifying how they were measured and their units.

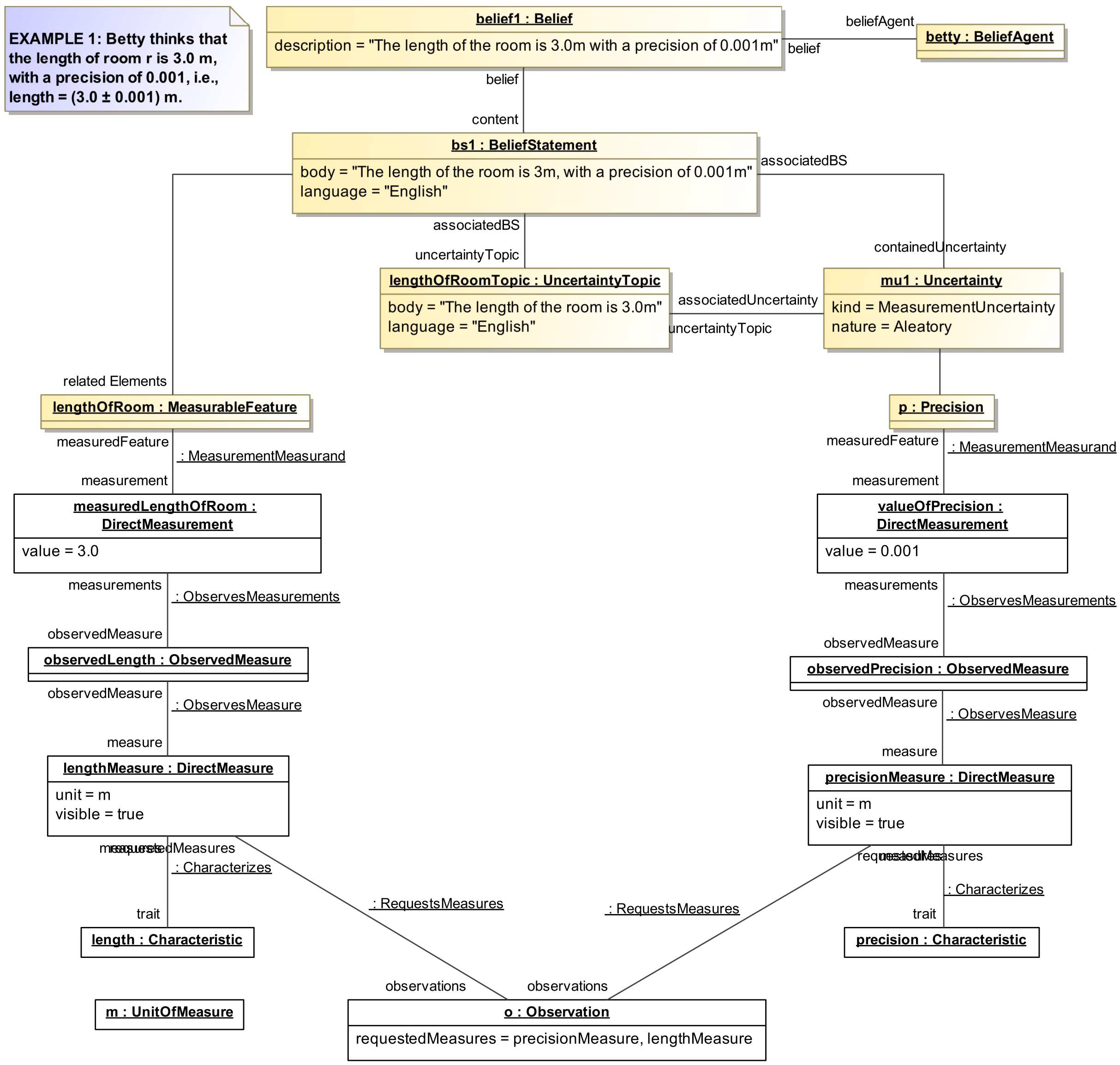

Figure A.2: Precision of Measurement Example

A.2.2 Adding Confidence to a Belief

A second example shows how to add a degree of belief to the previous example. The model below shows the specification of the statement:

"Betty thinks, with a confidence of 0.95, that the length of the room is 3m with a precision of 0.001m."

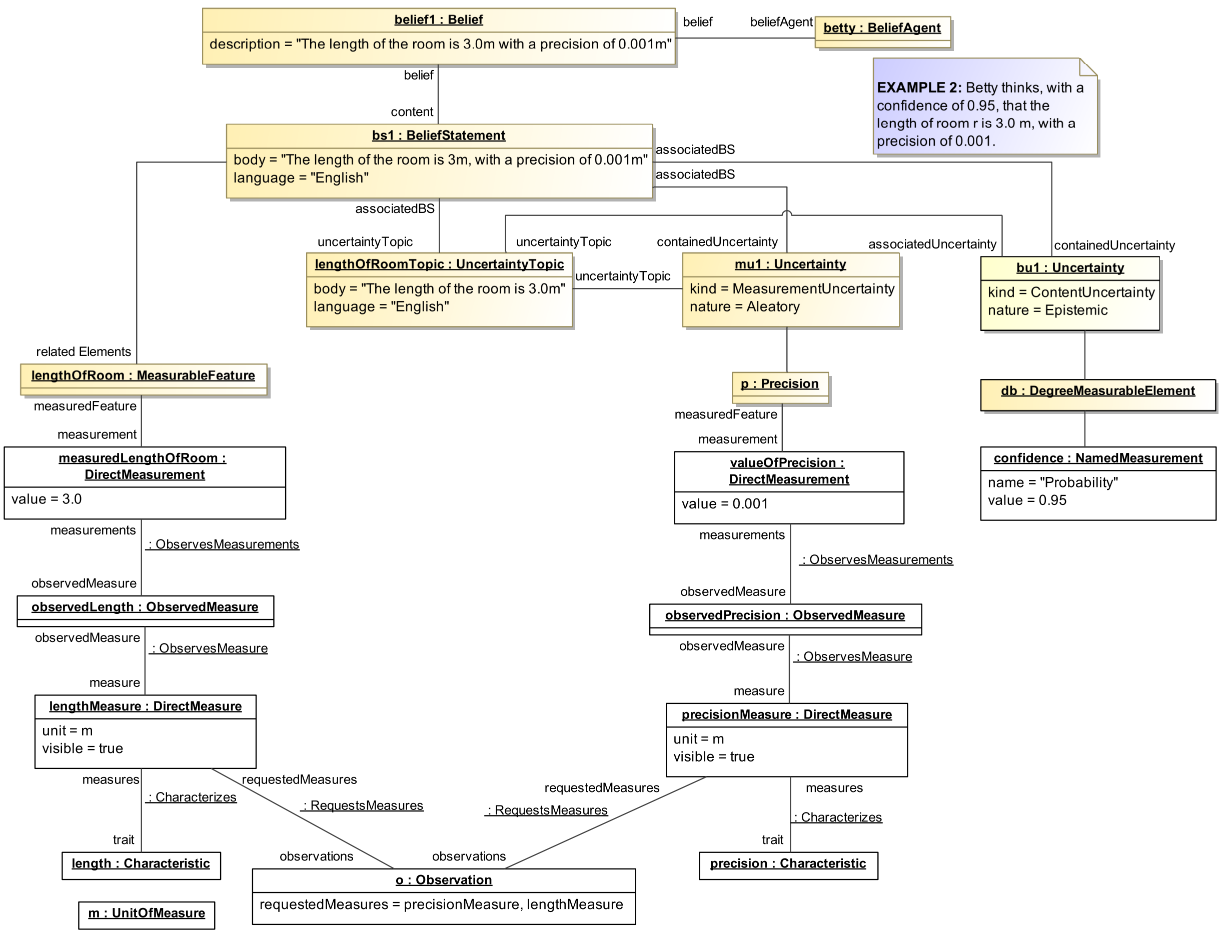

Figure A.3: Adding Confidence to Belief Example

Again, the unshaded boxes represents the SMM instances that specify the value of the measurements, how they are measured, and their units.

A.2.3 Working with two or more Belief Statements

The third example shows how to add two measurement uncertainties to a value. The model below shows the specification of the statement:

"Betty thinks that the length of the room is 3m with a precision of 0.001m and an accuracy of 0.002m."

Here the Belief is expressed using two separate *BeliefStatement*s.

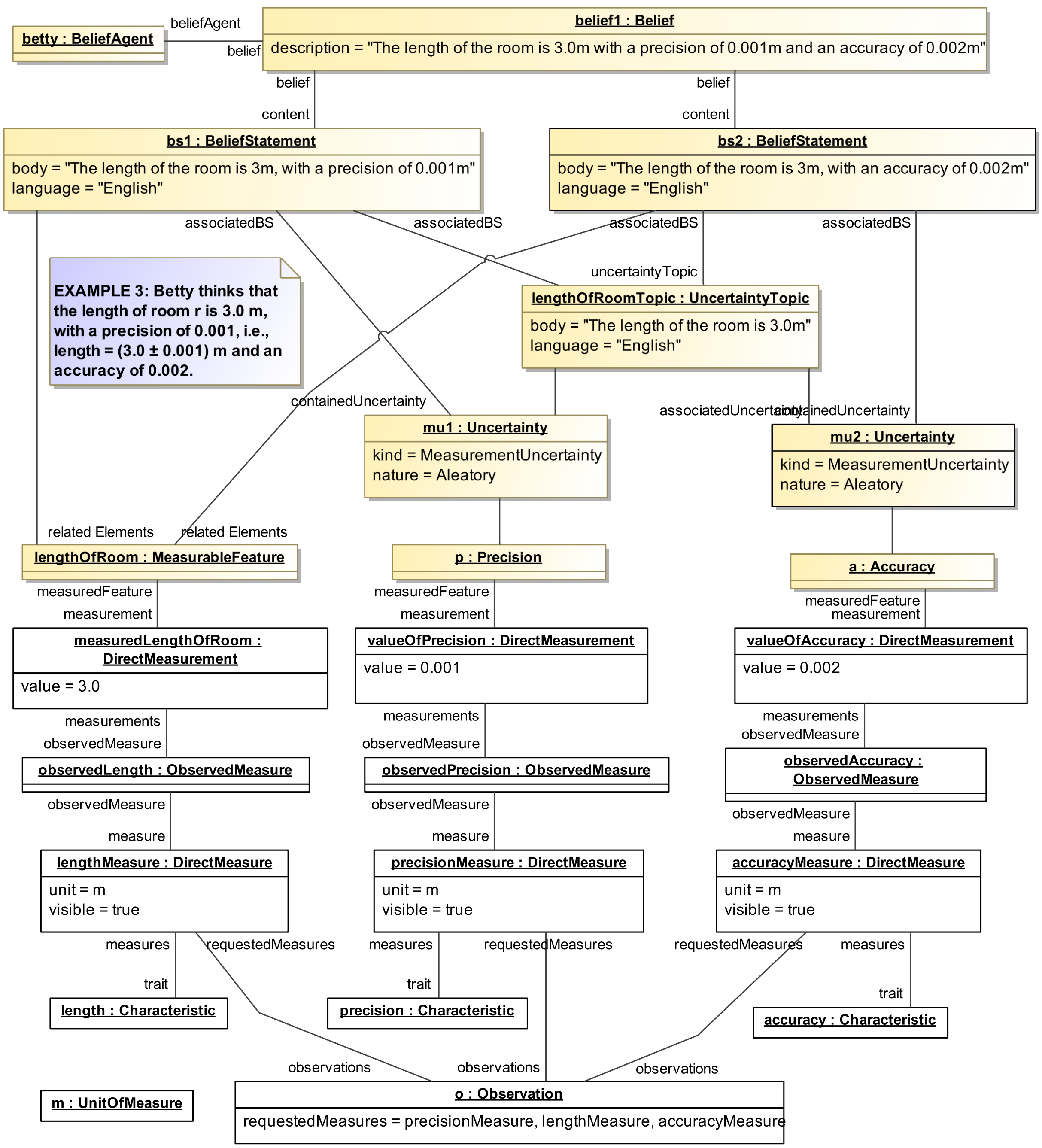

Figure A.4: Working with multiple Belief Statements Example

A.2.4 Assigning Uncertainty to Uncertainty

This example shows how to add uncertainty to a level of uncertainty already specified. The model below shows the specification of the statement:

"Betty thinks that the length of the room is 3.0m with a precision of 0.001m, and that the accuracy of the measurement of the precision is 0.002m."

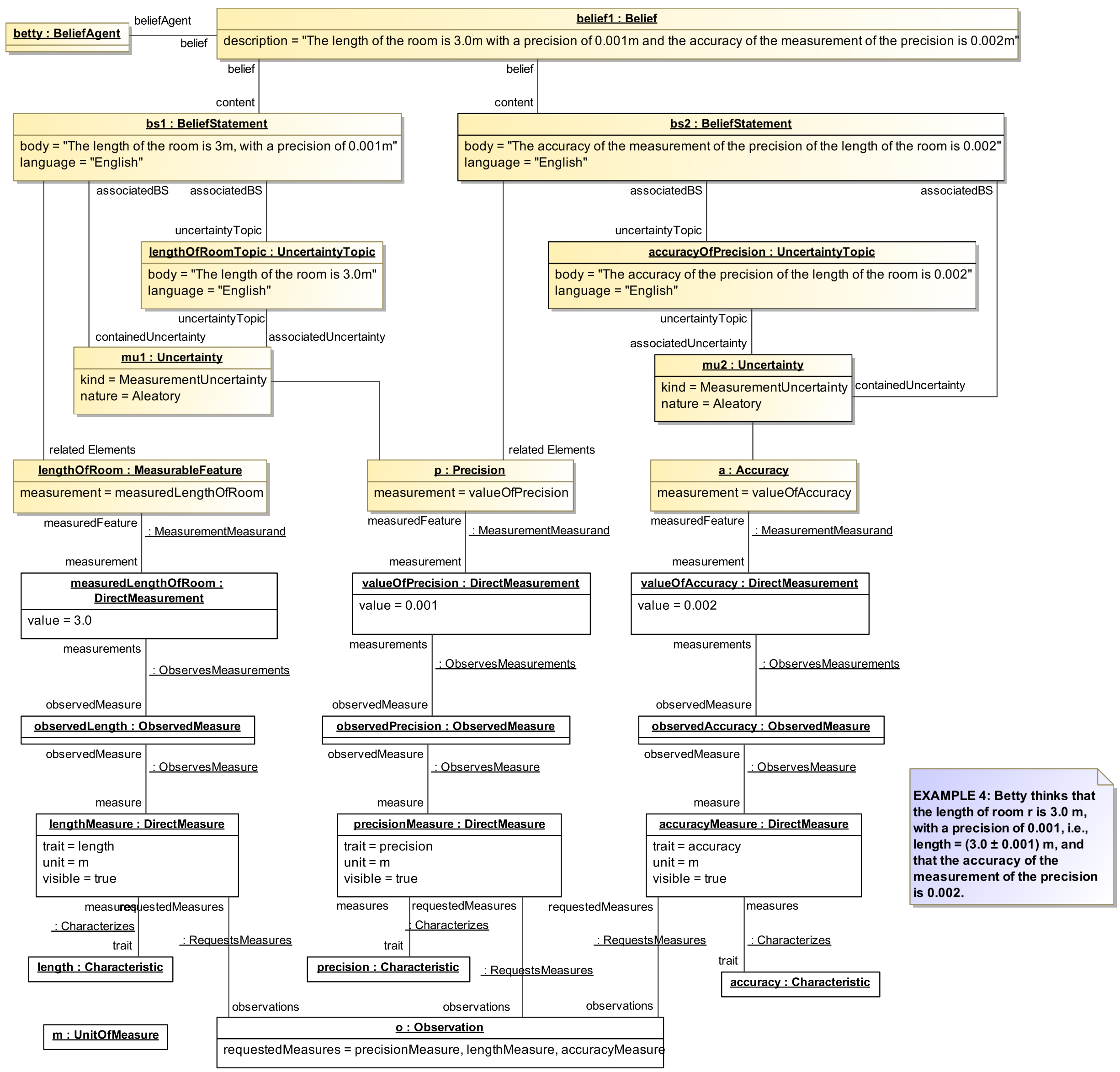

Figure A.5: Assigning Uncertainty to Uncertainty Example

Notice the difference with the previous example, where both precision and accuracy were associated to the same measurement (the length of the table is 3m), whilst here we are associating the accuracy to the measurement of the precision.

A.3 Operational Uncertainty due to Inconsistent Information

Our third example takes an operational perspective on uncertainty faced from the point of view of a software-intensive system during operation [16]. During the development of software-intensive systems, it is important to consider and understand the characteristics of the environment (or context) in which the system under development is supposed to operate. To that end, the operational context is analyzed and modeled. Especially in early development phases, e.g., requirements engineering, it is important to identify and understand the uncertainty that may occur when the system is in operation so that it can be equipped with capabilities to remain functional.

One situation that may occur especially in embedded and cyber-physical systems (e.g., in the automotive domain) is inconsistent information provided by different sensors monitoring the context. Autonomous vehicles process a broad range of context information. Such context information is acquired either through sensors or via communication between collaborating vehicles. The following object diagram shows uncertainty faced by some controller software systems embedded in a car, which needs to consolidate information obtained from different sources.

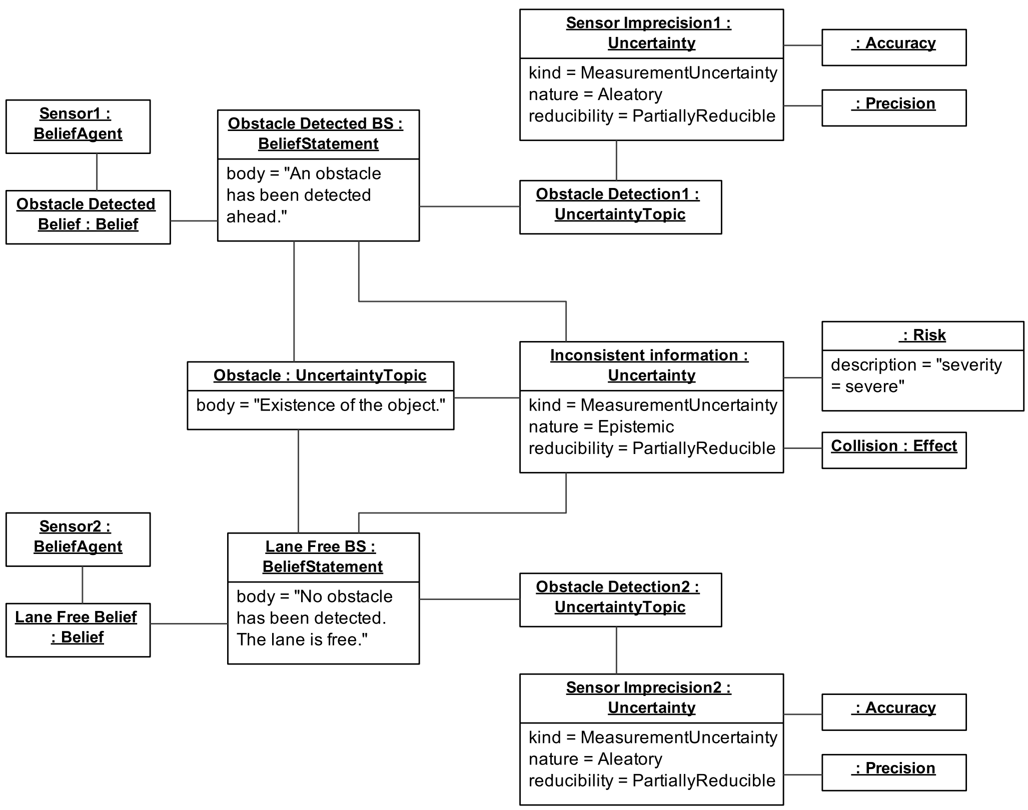

Figure A.6: Operational Uncertainty Example

In this example, there are two redundant onboard sensors installed in a vehicle (e.g. a Radar and a Lidar sensor). Both sensors monitor the road and are utilized to detect obstacles (such as other cars) ahead of the vehicle under consideration. These sensors are thus modeled as *BeliefAgent* instances. Given the measurements obtained from these sensors, *Belief* instances characterizing the current state of the vehicle's surroundings can be derived. Consider the situation of inconsistent *Belief* instances about the road ahead: The measurement obtained from *Sensor1* indicates that there is an obstacle (*Obstacle Detected Belief*) while the other sensor's (*Sensor2*) measurements do not (modeled as the *Lane Free Belief* instance). Using the *Belief* part of PSUM, we capture respective representations of these two beliefs as instances of the *BeliefStatement* concept. Note that we consider these pieces of information on a conceptual level without considering its concrete form, i.e., in a language-independent way.

Both *BeliefStatement*s on their own are subject to uncertainty due to the inherent imprecision of the two sensors. We connect the two belief statements to respective *Uncertainty* instances using the two *UncertaintyTopic* instances *Obstacle Detection1* and *Obstacle Detection2*. These uncertainties are both categorized as *MeasurementUncertainty* and can be quantified when considered as *MeasurableElements* using the *Measurement* part of PSUM. That way, it is possible to associate, e.g., *Precision* and *Accuracy* measurements. Since our previous examples have already elaborated in detail on capturing measurement uncertainty (see Annex A.2), we do not go into more details here. The two *Uncertainty* instances *Sensor Imprecision1* and *Sensor Imprecision2* are characterized as aleatory, as they are caused by physical aspects of sensor technology and the context. Furthermore, they are considered partially reducible, because, in principle, obtaining more information, such as through other, redundant sensors, helps reduce the uncertainty.

Nevertheless, merging information from two different sources also introduces another *Uncertainty* instance. From the point of view of the controller software during operation, which processes the two pieces of information in order to react by e.g. braking, this additional uncertainty relates to both *BeliefStatements* in combination. This is captured in the model through the additional *Uncertainty* instance *Inconsistent Information*. This uncertainty is connected to the two *BeliefStatements* mentioned above using *UncertaintyTopic* instances, and explicitly refers to both these statements in combination. The uncertainty topic can thus be phrased as “Existence of object”. The associated uncertainty is categorized as *Epistemic* because it can be considered on a higher, less technical level compared to the other two measurement uncertainties. The focus is on comparing two contradicting pieces of information rather than on the originating sources of this information. However, this uncertainty is also only partially reducible since the measurement uncertainty related to the original *Belief* instances remains.

The model also contains elements characterizing the *Risk* and *Effect* of the uncertainty. This is especially important when developing safety-critical systems such as embedded software in a vehicle, as in our example. There is a severe *Risk* instance that a *Collision* with an obstacle may occur as an *Effect*, in case the obstacle really exists. Thus, PSUM helps conceptualize and analyze uncertainty that may potentially occur during operation. From a software engineering perspective, domain-specific languages can be designed based on PSUM to support engineers in systematically building countermeasures into the system so that it will be able to cope with the defined uncertainty autonomously during operation.

Bibliography

[1] ISO. ISO 31000: Risk management. International Organization for Standardization (ISO), 2009.

[2] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. Uncertainty-wise cyber-physical system test modeling. *Software & Systems Modeling*, 18(2):1379–1418, 2019.

[3] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[4] Robert O Keohane, Melissa Lane, and Michael Oppenheimer. The ethics of scientific communication under uncertainty. *Politics, Philosophy & Economics*, 13(4):343–368, 2014.

[5] Amy R Bland and Alexandre Schaefer. Different varieties of uncertainty in human decision-making. *Frontiers in neuroscience*, 6:85, 2012.

[6] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. Understanding uncertainty in cyber-physical systems: a conceptual model. In *European conference on modelling foundations and applications*, pages 247–264. Springer, 2016.

[7] Javier Troya, Nathalie Moreno, Manuel F Bertoa, and Antonio Vallecillo. Uncertainty representation in software models: a survey. *Software and Systems Modeling*, 20(4):1183–1213, 2021.

[8] Javier Cámara, David Garlan, Won G Kang, Wenxin Peng, and Bradley Schmerl. Uncertainty in self-adaptive systems: Categories, management, and perspectives. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA, 2017.

[9] OMG. Structured Assurance Case Metamodel (SACM). 2015.

[10] OMG. Structured Metrics Metamodel (SMM) Version 1.2. 2018.

[11] OMG. Unified Modeling Language (UML) Version 2.5. 2015.

[12] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[13] JCGM. JCGM 200: International vocabulary of metrology - Basic and general concepts and associated terms (VIM). ed, 2012. Also, ISO/IEC Guide 99:2007.

[14] Man Zhang, Tao Yue, Shaukat Ali, Bran Selic, Oscar Okariz, Roland Norgre, and Karmele Intxausti. Specifying uncertainty in use case models. *Journal of Systems and Software*, 144:573–603, 2018.

[15] Tao Yue, Lionel C Briand, and Yvan Labiche. Facilitating the transition from use case models to analysis models: Approach and experiments. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 22(1):1–38, 2013.

[16] Torsten Bandyszak, Marian Daun, Bastian Tenbergen, Patrick Kuhs, Stefanie Wolf, and Thorsten Weyer. Orthogonal uncertainty modeling in the engineering of cyber-physical systems. *IEEE Transactions on Automation Science and Engineering*, 17(3):1250–1265, 2020.